**(1) Marked-up version**

**[0022]** In certain embodiments discussed below, the basic volume rendering algorithm that is applied to the data types mentioned above is commonly called "perspective raycasting." Starting from a virtual observer's eye, the volumetric data is visualized by shooting a viewing ray through the data set for each pixel on a virtual screen. Basically stated, each ray represents a sampling of points through the volume set, but these sampling points or "raypoints" in general do not lie on the data set grid; that is to say, a voxel represents a single color, intensity and opacity assigned to the smallest visual element in the data set. In a digital world, voxels are essentially average data assigned to a small region of space, and the raypoints may lie anywhere within these regions. Since two adjacent voxels may differ in color, each raypoint is typically computed by averaging the values of adjacent voxels. It is therefore typically necessary to approximate a value of each raypoint by averaging values from surrounding voxels, which is usually done by a process called "tri-linear interpolation." This process essentially assumes every raypoint lies on or somewhere in between two adjacent voxels in each one of *X, Y* and *Z* directions, i.e., within a cube consisting of eight voxels; the raypoint lying in this space is calculated by essentially averaging the values of the eight voxels in dependence upon how close the raypoint is to each voxel's center. The precise nature of processing of a raypoint depends on the particular voxel format, since each of the formats mentioned above generally has similar but distinct processes. The common goal of these processes, however, is to assign each raypoint a color and an opacity value. The color and opacity values of all raypoints on a given ray are then "composited" together using simplified absorption models that determine whether individual raypoints are opaque and, if not, how much they influence raypoints behind, to give the final pixel color. A large number of variants to this basic algorithm have been developed over time.

**(2) Clean version**

**[0022]** In certain embodiments discussed below, the basic volume rendering algorithm that is applied to the data types mentioned above is commonly called "perspective raycasting." Starting from a virtual observer's eye, the volumetric data is visualized by shooting a viewing ray through the data set for each pixel on a virtual screen. Basically stated, each ray represents a sampling of points through the volume set, but these sampling

points or "raypoints" in general do not lie on the data set grid; that is to say, a voxel represents a single color, intensity and opacity assigned to the smallest visual element in the data set. In a digital world, voxels are essentially average data assigned to a small region of space, and the raypoints may lie anywhere within these regions. Since two adjacent voxels may differ in color, each raypoint is typically computed by averaging the values of adjacent voxels. It is therefore typically necessary to approximate a value of each raypoint by averaging values from surrounding voxels, which is usually done by a process called "tri-linear interpolation." This process essentially assumes every raypoint lies on or somewhere in between two adjacent voxels in each one of *X, Y* and *Z* directions, i.e., within a cube consisting of eight voxels; the raypoint lying in this space is calculated by essentially averaging the values of the eight voxels in dependence upon how close the raypoint is to each voxel's center. The precise nature of processing of a raypoint depends on the particular voxel format, since each of the formats mentioned above generally has similar but distinct processes. The common goal of these processes, however, is to assign each raypoint a color and an opacity value. The color and opacity values of all raypoints on a given ray are then "composited" together using simplified absorption models that determine whether individual raypoints are opaque and, if not, how much they influence raypoints behind, to give the final pixel color. A large number of variants to this basic algorithm have been developed over time.

Please replace paragraph 26, found on page 8 of Applicant's specification with the following replacement paragraph.

**(1) Marked-up Version**

**[0026]** This operation is explained with reference to FIG. 1. In particular, a main memory (not depicted in FIG. 1) houses a data set that should be assumed to be of significant size, and hence, typically too large to be stored on processor or to be very easily managed. As indicated by function block 11, this data [[~~may~~]] set may represent data that is stored using a spread memory layout. Otherwise stated, conventional wisdom would call for a given data set to be stored as a contiguous group of data, in a contiguous space in memory. The present invention departs from this conventional wisdom and instead calls for breaking the

data set up into subsets of data having gaps in memory address in between. This operation may significantly increase the amount of "main memory" required to house the data set, but it will facilitate storing and processing efficiencies that will assist processing by conventional processors, as will be seen below. The precise size of the gaps will also be further discussed below, but it should be noted that the purpose of this spread memory layout is to result in data inherently being addressed such that individual datum or small groups of data may be retrieved and stored in a particular location within quick access memory used by the processor, so as to not overwrite needed processing parameters.

**(2) Clean Version**

**[0026]** This operation is explained with reference to FIG. 1. In particular, a main memory (not depicted in FIG. 1) houses a data set that should be assumed to be of significant size, and hence, typically too large to be stored on processor or to be very easily managed. As indicated by function block 11, this data set may represent data that is stored using a spread memory layout. Otherwise stated, conventional wisdom would call for a given data set to be stored as a contiguous group of data, in a contiguous space in memory. The present invention departs from this conventional wisdom and instead calls for breaking the data set up into subsets of data having gaps in memory address in between. This operation may significantly increase the amount of "main memory" required to house the data set, but it will facilitate storing and processing efficiencies that will assist processing by conventional processors, as will be seen below. The precise size of the gaps will also be further discussed below, but it should be noted that the purpose of this spread memory layout is to result in data inherently being addressed such that individual datum or small groups of data may be retrieved and stored in a particular location within quick access memory used by the processor, so as to not overwrite needed processing parameters.